Relieving Hot Spots in Collaborative Intrusion Detection Systems during Worm Outbreaks

Chenfeng Vincent Zhou, Shanika Karunasekera and Christopher Leckie

NICTA Victoria Research Laboratory
Department of Computer Science and Software Engineering,
The University of Melbourne, Australia

Email: {cvzhou, shanika, caleckie}@csse.unimelb.edu.au

***Abstract*—The increasing number of stealthy and coordinated attacks on the Internet pose a significant threat to network security. Collaborative intrusion detection systems (CIDSs) have therefore been proposed to address this coordinated defense challenge by correlating patterns of suspicious activity based on the source addresses of the suspicious incoming traffic. However, during worm outbreaks, there can be a rapid growth in suspicious evidence that is reported about individual sources of the worm outbreak. In CIDSs that correlate suspicious activity by source address, the evidence relating to these worm spread sources can cause a load "hot-spot", which severely degrades the overall performance of the detection system. In this paper, we propose a load balancing scheme for a CIDS to evenly distribute the workload to avoid hot-spots during worm outbreaks. Rather than correlating suspicious evidence based on source addresses, we distribute the load in the CIDS using a scheme that enables different possible patterns of suspicious evidence to be automatically mapped onto different processing nodes in the CIDS. Simulation results show that our scheme can achieve significant improvements in load balancing without sacrificing detection accuracy.**

I. Introduction

When coordinated attacks such as distributed denial of service (DDoS), worms and coordinated scans occur simultaneously across multiple networks, these attack activities may initially appear innocent at each local network, as the evidence for an attack may be distributed across a large number of networks. In response to this problem, collaborative intrusion detection systems (CIDSs) have been proposed to address this coordinated defense challenge [1], [2], [3]. In a CIDS, participating IDSs exchange evidence on suspicious incoming traffic that has been collected locally, from which a global decision can be made. The CIDS is hence able to obtain a global view of the intrusion activity, without all IDSs conforming to a single scheme. In addition, by correlating information from multiple networks, potentially suspicious activities can be detected faster than if each participating IDS operates in isolation. However, the design of a robust CIDS raises several research challenges, which can be summarized as follows: (1) How to remove the need for a central controller in the CIDS; (2) How to balance the trade-off between the detection rate and false alarm rate in the CIDS; (3) How to improve the scalability of alert correlation; (4) How to achieve local balancing across the CIDS, so that no individual participating IDS becomes overloaded.

In our previous work, we have proposed a peer-to-peer (P2P) based decentralized collaborative intrusion detection system to address the scalability problem of a CIDS [3]. We presented an optimization scheme to balance the trade-off between the detection rate and false alarm rate based on an analysis of a large, real-world intrusion dataset [4]. We also proposed a "correlate-and-filter" algorithm to analyze multi-dimensional alerts in a CIDS, based on a lattice of possible patterns that correspond to specific categories of attacks [5]. In our approach, we introduced a hierarchical two-stage scheme to distribute the computation in the CIDS, by analyzing raw alerts locally at each IDS before forwarding pattern instances to a central server for global analysis. We also presented a fully decentralized CIDS architecture, based on a P2P publish/subscribe protocol, to support our two-stage scheme. The proposed decentralized CIDS architecture eliminates the need for a centralized server. However, it can generate load hot spots among participant IDSs, when all the suspicious traffic comes from the same source during a very short time span, *e.g.*, during a worm outbreak. This occurs when an infected victim machine starts to infect others by massively scanning the whole network to look for a specific software vulnerability. In this scenario, the alert information reported by all participant IDSs will involve the same traffic source. As a result, all these alerts will be assigned to the same participating node that is responsible for this source address in the CIDS. This node will become a load "hot spot" in the CIDS, which may cause delays in correlation or even information loss. Moreover, an attacker can exploit this vulnerability by launching massive scans to multiple networks from a single source in order to overload the responsible participating IDS, and ultimately disrupt the CIDS.

In this paper, we propose a load balancing scheme for CIDSs, which evenly distributes work load during worm outbreaks, hence addressing the last challenge stated above. Maximizing the detection rate (*i.e.*, detection coverage) and minimizing the false positive rate are the common goals of IDS research. However, it is essential to note that an IDS also needs to be resistant to attack in order to be effective. Our focus on this paper is to relieve the load "hot spot" in the

978-1-4244-2066-7/08/$25.00 ©2008 IEEE

CIDS to improve the performance and robustness of the whole CIDS without significantly sacrificing its detection accuracy.

The remainder of this paper is structured as follows. In Section II, we present the load hot spot problem. By analyzing real world intrusion data from a worm outbreak as well as a normal stealthy scan scenario, we demonstrate how a load hot spot is created in a CIDS. In Section III, we propose a pattern mapping approach to balance the work load in a CIDS during worm outbreaks, and then describe the underlying algorithm behind this scheme. In Section IV, we evaluate our load balancing scheme in terms of load distribution and detection accuracy using a simulated worm outbreak dataset and a real word scan dataset. We discuss the related work in Section V. Finally, we outline future work and conclude in Section VI.

II. Problem Statement

Fully distributed CIDS architectures have been proposed to address the scalability problem that arises in a centralized CIDS. In a fully distributed CIDS, each participant IDS has two functional units: a data unit that is responsible for collecting data locally, and a correlation unit that is a part of the distributed correlation scheme. The participant IDSs communicate with each other using a distributed protocol, such as P2P [1], [2], [3], [4], [5], [6], gossiping [7], multicast or publish/subscribe protocols [8], [9].

Most CIDSs correlate suspicious evidence based on the source addresses of the suspicious traffic sources [2], [3], [4], [5], [8]. These systems consider the source address as the key attribute for correlation because most coordinated attacks such as worms or coordinated scans share the same source IP address, and many attack signatures include this source address as a field. However, these CIDSs that are based on source address correlation are prone to generate a highly skewed load distribution among participating IDSs during worm outbreaks when most of the suspicious traffic comes from the same source. This results in a single IDS being severely overloaded, which we call a load "hot spot".

In this section, we present the load balancing problem of hot spots in a decentralized CIDS [5]. We then investigate the behavior of suspicious source addresses in traffic traces of an actual worm outbreak and a stealthy scan, to demonstrate the type of traffic load that can create a hot spot in a CIDS. Although we use a specific example of a CIDS to explain the problem of load hot spots, this problem can occur in any CIDS that uses source addresses as the basis for correlation [3], [4], [5], [8].

A. Cause of Load Hot Spots in a Decentralized CIDS

We consider the problem of correlating multi-dimensional alert patterns, based on the raw alerts that are reported by a set of IDSs. In [5], we proposed a "correlate-and-filter" algorithm to address the problem of improving the scalability of alert correlation in a CIDS. In this approach, we constrain the search space for multi-dimensional alert patterns by using knowledge of the types of attack categories of interest to limit the search to certain predefined combinations of dimensions or features. In particular, redundant alerts are avoided by suppressing a more general alert pattern instance P_1 if there is a more specific pattern instance P_2 that accounts for significant proportion of the evidence for the general pattern instance P_1. Our "correlate-and-filter" algorithm uses a hierarchical two-stage scheme to distribute the computation in the CIDS. The raw alerts are analyzed locally at each IDS before being forwarding to a central server for global analysis. A P2P based CIDS architecture was proposed to support this two-stage scheme, where the alert correlation is performed in a publish/subscribe manner. The proposed decentralized CIDS architecture eliminates the need for a centralized server. However, a load hot spot can be created when a small number of source addresses generate a significant proportion of the suspicious activities.

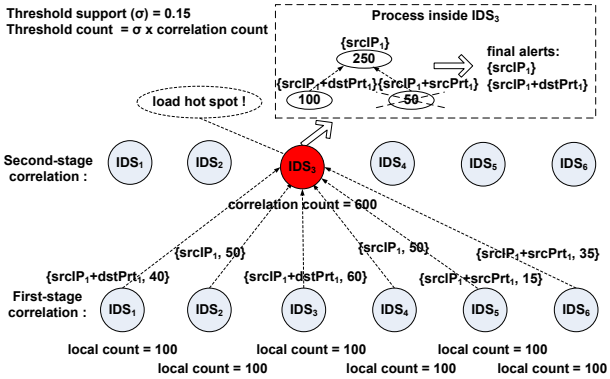


Fig. 1. Decentralized CIDS architecture based on source address correlation

As shown in Figure 1, consider the example where six IDSs are participating in the CIDS, which uses the "correlate-and-filter" algorithm for alert correlation. At the end of monitoring time interval Δ, each IDS_i collects suspicious incoming evidence from its local subnetwork. This evidence is in the form of a *raw alert*, where an alert is the standard 5-tuple *<srcIP,srcPrt,dstIP,dstPrt,prot>*, and *srcIP* is the source IP address of the connection, *srcPrt* is the source port, *dstIP* is the destination IP address, *dstPrt* is the destination port, and the *prot* is the protocol used. A pattern is a subset of these five features. The raw alerts are processed locally by running the "correlate-and-filter" algorithm to find instances of suspicious patterns of evidence with high support, where the *support* of a pattern *P* is defined as the number of raw alert records that match pattern *P*, as a proportion of the total number of raw alerts received [1].

Consider an example where each participating IDS receives 100 raw alerts, given the support threshold σ = 0.15, *i.e.*, the local threshold count is 15 alerts. Figure 1 also shows all significant pattern instances generated by each IDS. For example, since the count of pattern instance $\{srcIP_1 + dstPrt_1\}$ (40) exceeds the local threshold (15), this instance is considered as a locally significant pattern by IDS_1. This pattern is then reported to the CIDS to be correlated with locally significant patterns from other participants in order to make a consensus decision. In more detail, IDS_1 subscribes to the pattern instance $\{srcIP_1 + dstPrt_1\}$ in the CIDS, which is implemented by mapping the instance along with its count to

[1]Please refer to [5] for detail of the algorithm.

a participating IDS using consistent hashing [10], based on the source address of the pattern instance, *i.e.*, $srcIP_1$. In this example, IDS_3 is responsible for hosting all the subscriptions related to source address $srcIP_1$. In order to estimate the *correlation count* at the global level, the total count of raw alerts on IDS_1 (100) will also be sent to IDS_3. Likewise, after performing the "correlate-and-filter" algorithm locally, IDS_2 reports pattern instance $\{srcIP_1\}$ along with its count (50) and local total count (100) to IDS_3 for correlation; IDS_3 reports pattern instance $\{srcIP_1 + dstPrt_1\}$ along with its local count (60) and local total count (100) on itself, and so on for each IDS. After IDS_3 receives all the suspicious evidence regarding $srcIP_1$, the correlation count comes to 600 since all six participants have reported evidence relating to $srcIP_1$ to IDS_3. Hence the threshold count for the second correlation stage on IDS_3 is 90. Then IDS_3 applies the "correlate-and-filter" algorithm over all $srcIP_1$ related suspicious evidence reported by participating IDSs. Since the total count of pattern instance $\{srcIP_1 + srcPrt_1\}$ does not exceed the threshold count of 90, this pattern instance is pruned. However, since the total count of pattern instance $\{srcIP_1 + dstPrt_1\}$ exceeds the threshold count, this pattern instance is considered as significant. Therefore, the final alerts are pattern instances $\{srcIP_1\}$ and $\{srcIP_1 + dstPrt_1\}$ [2].

This example demonstrates how source address based correlation is able to correlate suspicious evidence by source address. Consequently, the second stage "correlate-and-filter" algorithm can be performed in a fully decentralized manner, *i.e.*, each participating IDS at the second stage is able to make its decision independently, since all the alerts related to a certain source address will be reported to the same node. However, if a source address generates traffic that is reported by a large number of participating IDSs, that source address can create a load hot spot on the participating node that is responsible for correlating evidence pertaining to this source address. In this example, since all the participants report evidence related to the same source address $srcIP_1$, the participant that is responsible for correlating evidence about this particular source address will become a load hot spot.

B. Example of Load Hot Spots Arising in Real World Intrusion Datasets

We demonstrate how load hot spots can occur in practice based on actual intrusion traces for a real worm outbreak and a stealthy scan scenario.

1) Intrusion Trace Data: We use two intrusion datasets obtained from DShield.org [11] - a global repository of firewall/NIDS (Network Intrusion Detection System) logs for research purposes, which is a part of the Internet Storm Center project of the SANS Institute. The first dataset – the SQL-Slammer Dataset – is a dataset collected by DShield during the SQL-Slammer worm outbreak from January 15 2003 to January 27 2003 (port 1434/udp intrusion logs). We use this dataset to analyze suspicious source behavior during a worm attack. The second dataset – DShield Dataset – that is used in our test comprises a large set of firewall and NIDS logs collected from 1682 firewall/NIDS platforms all over the world for the period from 1-15 January 2005. There is no specific worm outbreak included in this time period. Therefore, we use this dataset as a source of stealthy scans and coordinated scans, to analyze the corresponding source behavior. There are 501,488,037 records in these logs, and the size of the whole dataset is more than 38.8 GB.

[2]According to the "correlate-and-filter" algorithm, $\{srcIP_1\}$ is considered as globally significant since its significant children pattern instances (*i.e.*, $\{srcIP_1 + dstPrt_1\}$) are not sufficient to explain all the alerts covered by $\{srcIP_1\}$, *i.e.*, the difference between the count for pattern instance $\{srcIP_1\}$ and the count for reported pattern instance $\{srcIP_1 + dstPrt_1\}$ exceeds the threshold count ($250 - 100 > 90$).

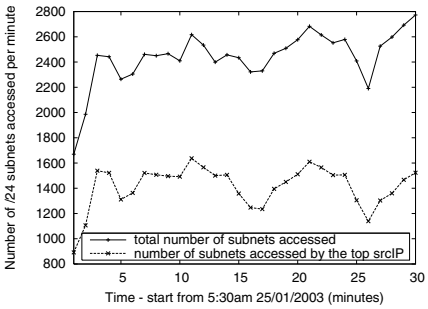


Fig. 2. Suspicious source behavior during SQL-Slammer outbreak (5:30-6:00am 25 January 2003)

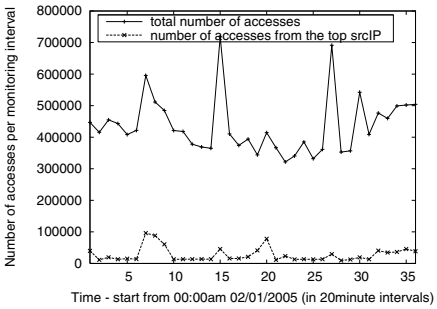


Fig. 3. Suspicious source behavior in a normal scan scenario (00:00am - 12:00pm 2 January 2005)

2) Source Behavior during Worm Outbreak: Assuming there is an IDS monitoring the incoming traffic of each /24 subnetwork, we plot in Figure 2 the number of /24 subnetworks that were accessed by the most active source address every minute during the peak period of the sql-slammer worm outbreak from 5:30am to 6:00am on January 25, 2003. As shown in Figure 2, one source address consistently scanned more than 50% of all monitored /24 sub-networks in every minute during the worm outbreak. Suppose the CIDS is operated during that time period, and one minute is set as the monitoring time interval Δ, then 50% of all the alerts reported during that time interval will be sent to a single node. As a result, a hot spot will be created on the participating node that is hosting the top source address within that time interval.

3) Source Behavior during Stealthy Scans: In order to investigate the source behavior during a normal scan scenario where there is no specific worm present, we plot the total number of accesses generated by the top source address in every twenty minute period across twelve hours from 00:00am to 12:00pm on January 2, 2005, which was selected at random

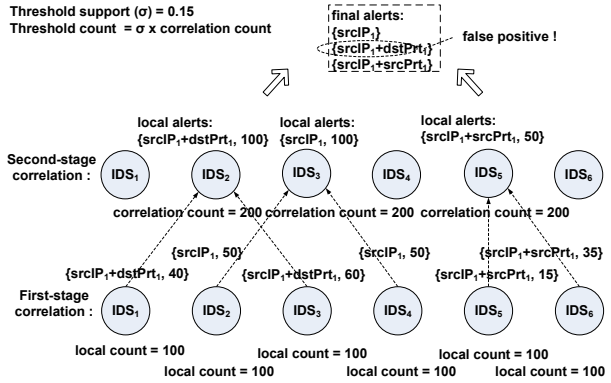


Fig. 4. Decentralized CIDS architecture based on pattern instances correlation

from the DShield dataset. As shown in Figure 3, the top source does not account for a significant proportion of all the accesses on average. However, during particular time periods, as shown in Figure 3, the top source address was responsible for around 17% of all accesses within the time period of 2:00 - 2:20am (intervals 6-7) and 6:20 - 6:40am (intervals 19-20) on 02/01/2005. If a CIDS based on source address mapping is used in this scan scenario, there will be a load surge on the node that is hosting the suspicious evidence for the top source address in the 2:00 - 2:20am or 6:20 - 6:40am intervals on the 2 January 2005.

III. Balance the Workload using Pattern based Correlation

In this section, we propose a load balancing scheme for a CIDS to evenly distribute the workload onto different processing nodes during worm outbreaks. We first use an example to describe our scheme, then present the algorithm underlying this scheme in the context of a decentralized CIDS.

A. Load Balancing Scheme

In order to relieve the workload on the hot spot node in the CIDS, we propose to distribute suspicious evidence by indexing according to each possible alert pattern instance. We use an example in Figure 4 to explain our approach. The first stage of this example is the same as the example in Figure 1, *i.e.*, there are six IDSs participating in the CIDS that are using the "correlate-and-filter" algorithm for alert correlation. At the end of the monitoring time interval Δ, each IDS_i collects suspicious evidence from its local subnetwork, then runs the "correlate-and-filter" algorithm locally to generate the alert pattern instances for correlation. The same threshold parameters are used as in the previous example. As shown in Figure 4, after the first stage correlation, local alert pattern instances are generated, *e.g.*, $\{srcIP_1 + dstPrt_1, 40\}$ on IDS_1, $\{srcIP_1, 50\}$ on IDS_2, *etc*. Instead of using the source address as the key for routing suspicious evidence to the appropriate peer node in the second stage, as was done in previous example, we map the evidence based on the specific pattern instance. For example, as shown in Figure 4, based on the consistent hashing scheme used in the routing decision for each pattern instance, IDS_2 is responsible for hosting pattern instance $\{srcIP_1 + dstPrt_1\}$ from IDS_1 and IDS_3, IDS_3 is responsible for pattern instance $\{srcIP_1\}$ from IDS_1 and IDS_3, and IDS_5 is responsible for hosting pattern instance $\{srcIP_1 + dstPrt_1\}$ from IDS_1 and IDS_3. Therefore, compared to the correlation scheme based on source addresses in the previous example, all the workload related to the same source address $srcIP_1$ is now distributed over three participants IDS_2, IDS_3 and IDS_5, rather than flooding one single node. The load hot-spot is hence removed in this example.

After applying the routing scheme based on pattern instances, the second stage correlation in Figure 4 is as follows. IDS_2, IDS_3 and IDS_5 receive alert reports from two participants respectively, hence their correlation count is 200 as shown in Figure 4. Consequently, the threshold count for second stage correlation is 30 in this example. Instead of applying the "correlate-and-filter" algorithm at the second stage, each IDS makes its decision based on the count of the pattern instances received, which eliminates the need to group all pattern instances that are related to the same source address together. Therefore, IDS_2 reports pattern instance $\{srcIP_1 + dstPrt_1\}$ as an alert, since its count exceeds the threshold count 30. Likewise, IDS_3 reports pattern instance $\{srcIP_1\}$ and IDS_5 reports pattern instance $\{srcIP_1 + srcPrt_1\}$ as alerts.

One concern of using this routing scheme based on pattern instances is the possible degradation in detection accuracy which is caused by the different correlation counts at each node in the second stage. As shown in Figure 4, the final reports of the CIDS contains $\{srcIP_1{+}dstPrt_1\}$, $\{srcIP_1\}$ and $\{srcIP_1{+}srcPrt_1\}$. Compared to the previous source address based correlation scheme, the instance $\{srcIP_1 + srcPrt_1\}$ is a false alarm. This false positive is created by the difference in threshold counts on each of the second stage correlation nodes, since the suspicious evidence is distributed among different nodes in the CIDS. Consequently, each node only has a partial view in terms of the correlation count, *i.e.*, the correlation count on IDS_2, IDS_3 and IDS_5 is 200 rather than 600. In reality, however, the number of pattern instances reported to the second stage are far greater than the number of participant IDSs, and since we report each pattern instance along with the total count of its source IDS, there is a high probability that the correlation nodes on the second stage can receive a reasonable estimate of the global correlation count. In Section IV, we demonstate the detection accuracy of our modified scheme using both simulated worm outbreak data and real world stealthy scan data.

B. Load Balancing Algorithm

We now describe the algorithm of our proposed load balancing scheme in the context of a decentralized CIDS which was initially proposed in [5].

As described in [5], each participating IDS d_i plays two roles: (1) a *detection* unit to be responsible for maintaining the suspicious evidence for its protected subnetwork, and (2) an *alert correlation* unit to be responsible for correlating subscription messages and generating notification messages about the suspicious evidence that are mapped to that participant. In

Algorithm 1 CIDS Load Balancing Algorithm

```
1: In detection system d_i:
2: // local process (first stage correlation)
3: for each time interval Δ do
4:     collect raw alerts r_i locally
5:     // support threshold σ is a system parameter
6:     // LA_i: local alert report on d_i
7:     LA_i ⇐ correlate-and-filter(r_i, σ)
8:     for each p_ij ∈ LA_i do
9:         // look up the destination node for p_ij
10:        d_t = lookup(p_ij)
11:        // c_ij is the count of pattern p_ij
12:        subscribe(p_ij, c_ij, |r_i|, d_i) on d_t
13:    end for
14: end for
15: // Subscription and Notification Process on d_t (second stage correlation)
16: while message received do
17:    if message == subscribe(p_ij, c_ij, |r_i|, d_i) then
18:        // Sub_t: subscription pattern list on d_t
19:        Sub_t ⇐ < p_ij, c_ij, d_i >
20:        // NodeLst_t: source node list on d_t
21:        NodeLst_t ⇐ < d_i, |r_i| >
22:        if last subscription within this time interval then
23:            C_t: correlation count on d_t
24:            C_t ⇐ get-count(NodeLst_t)
25:            for each p_ij ∈ Sub_t do
26:                if p_ij ≥ σ × C_t then
27:                    for each d_k who subscribed to p_ij do
28:                        notify(d_k, p_ij)
29:                    end for
30:                end if
31:            end for
32:        end if
33:    else if message == notify(d_t, p_ij) then
34:        p_ij is confirmed as an attack pattern instance
35:    end if
36: end while
```

the first role, each participant generates $subscribe$ messages, and receives $notify$ messages; while in the second role, each participant receives $subscribe$ messages and generates $notify$ messages.

As shown in Algorithm 1, The first stage correlation is almost identical to the algorithm described in [5], except for the message routing scheme. In brief, at the end of each monitoring time interval, the participating IDS d_i collects alerts r_i based on the incoming traffic to its local subnetwork. Then the local alert report $\mathsf{LA_i}$ is generated by running the "correlate-and-filter" algorithm over r_i using the threshold support σ. After that, each pattern instance p_{ij} in $\mathsf{LA_i}$ is subscribed to its destination node d_t using the pattern instance mapping scheme.

In the second stage correlation, when the subscribe message is received by d_t, the pattern instance p_{ij}, its count c_{ij} and its source IDS are stored in the subscription list $\mathsf{Sub_t}$ to prepare for correlation. In order to calculate the correlation count later, the source IDS d_i and its total count r_i are added to the source node list $\mathsf{NodeLst_t}$. After all the subscription messages generated in the current time interval Δ and assigned to d_t have been received, the correlation count $\mathsf{C_t}$ is calculated using the information in $\mathsf{NodeLst_t}$. Then the count of each pattern instance p_{ij} in the subscription list $\mathsf{Sub_t}$ is compared to the threshold count ($\sigma \times C_t$), and each subscriber of p_{ij} will be notified if the total count of p_{ij} exceeds the threshold count. If a notification message of pattern instance p_{ij} is received by d_t, then p_{ij} is confirmed as an attack pattern instance.

IV. Evaluation

In order to evaluate the effectiveness of the proposed load balancing scheme during a worm outbreak, we simulate the CIDS with our load balancing scheme as described in Section III. We use a simulated worm outbreak dataset in the simulation, since the sql-slammer dataset as described in Section II only consists of the attack traffic (*i.e.*, it contains no other types of alerts), which is insufficient for this evaluation. We measure the load distribution on the participating nodes as well as the detection accuracy compared to the original source address based correlation scheme. We also evaluate the detection accuracy of the CIDS using our pattern instance correlation scheme based on data from real world stealthy scans where there is no specific worm outbreak.

A. Test Datasets

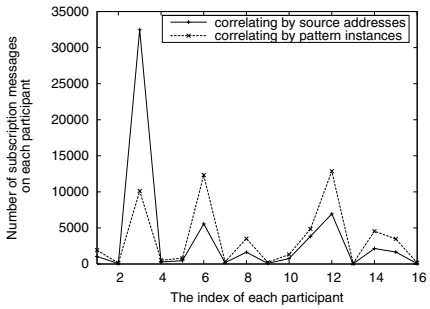
There are two datasets used in this simulation.

1) **Simulated worm outbreak dataset**: This dataset is generated using the DShield stealthy scan dataset as described in Section II by modifying the *srcIP* field of alert entries to create a specific source IP address that is responsible for 50 percent of the scan traffic over the monitoring time interval Δ. There are two advantages of our simulated dataset. First, the source behavior in this simulated dataset is statistically similar to the behavior of the scanning source during the sql slammer worm outbreak as shown in Figure 2. Furthermore, other fields in each alert entry such as *prot*, *srcPrt* and *dstPrt*, come from the real world intrusion dataset.
2) **Real world stealthy scan dataset**: We select an intrusion log for a randomly selected day from the DShield stealthy scan dataset as described in Section II.

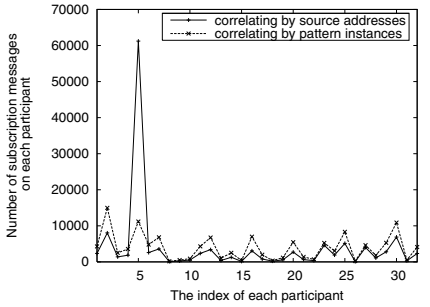
B. Evaluation Metrics

We use the following metrics to measure the effectiveness of our proposed load balancing scheme.

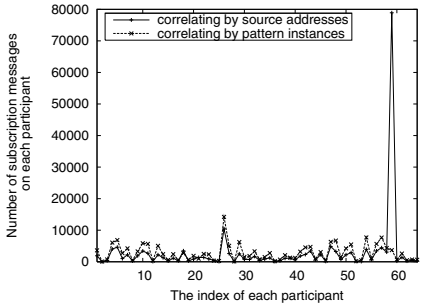
1) **Load distribution**: We record the number of subscription messages on each participant.
2) **Detection accuracy**: We compare our proposed pattern instance correlation scheme to the decisions made by the previous source address based scheme.
   a) **Detection rate**, *i.e.,*
   $$DR = \frac{\#True\ Positive}{\#True\ Positive\ +\ \#False\ Negative}.$$
   The detection rate may be affected by incorrect generalizations since the "correlate-and-filter" algorithm is not performed on the second stage correlation in our load balancing scheme.
   b) **False positive rate**, *i.e.,*
   $$FPR = \frac{\#False\ Positive}{\#False\ Positive\ +\ \#True\ Negative}.$$



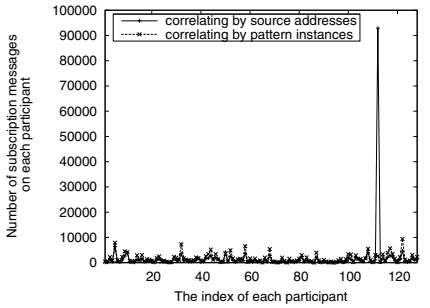
(a) 16 participants



(b) 32 participants



(c) 64 participants



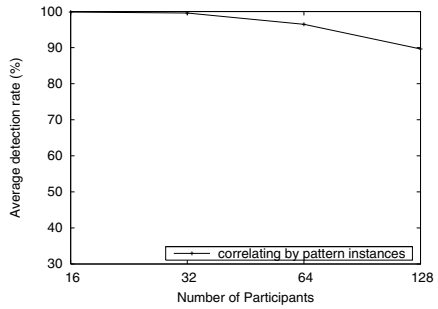
(d) 128 participants

Fig. 5. Evaluation of the load-balancing scheme by simulation

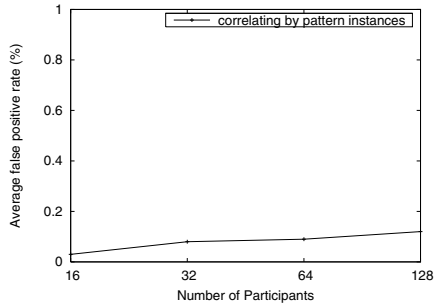
A false positive may occur as a result of the different correlation count in second stage correlation as described in Section III, or as a result of a pattern instance having the incorrect level of generalization.

C. Simulation Results during Worm Outbreaks

In this simulation, we applied our proposed approach to the simulated worm outbreak dataset, by varying the number of participating nodes from 16 to 128. Figure 5 plots



(a) Detection rate of pattern correlation scheme



(b) False positive rate of pattern correlation scheme

Fig. 6. Evaluation of detection accuracy of the pattern correlation scheme during worm outbreak

the distribution of the subscription messages among all the participants. The pattern correlation scheme in each case is able to distribute the subscription messages evenly among participants. In contrast, there is an uneven load distribution in the source address based correlation scheme, where a single node is responsible for most of the worm alerts. As shown in Figure 5, as the number of participants increase, the total number of subscription messages increases as well. For example, in the case of 128 participating IDSs (Figure 5(d)), the load on the most heavily loaded participant using the source correlation scheme is reduced by a factor of around 10 when our proposed load balancing scheme is used, *i.e.*, a 90% reduction in peak load. In the source correlation scheme, the increased number of subscription messages containing the worm outbreak source (*i.e.,* the simulated *srcIP* in the simulation) are all mapped to the same participant. Therefore, the load hot spot becomes worse when we increase the number of participants. In contrast, the pattern instance correlation scheme is able to divide the increased load of alerts containing the worm outbreak source into different patterns, then map them to different participants in the CIDS. Furthermore, with the increase in the number of participants and different pattern instances, the underlying consistent hashing function that is used to route alerts is able to achieve better load balancing [10]. Therefore, the pattern correlation scheme is able to distribute the subscription messages more evenly among participants when we increase the number of participants.

Figure 6 plots the evaluation results in terms of detection accuracy. Figure 6(a) shows the detection rate of the proposed load balancing scheme compared to the previous source address correlation scheme. In general, the detection rate of our new pattern instance based correlation scheme is more than

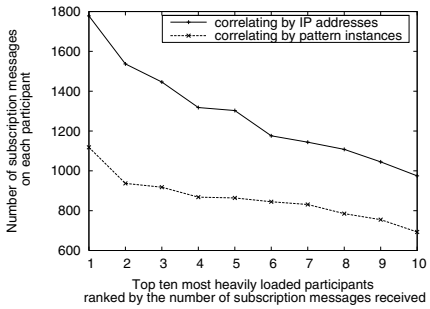


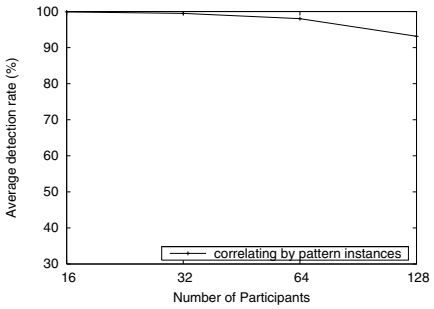
Fig. 7. Evaluation of our load-balancing scheme on the real world stealthy scan dataset

90%. There is a slight decrease in the detection rate from 100% to 90% when we increase the number of participants from 16 to 128, which is possibly caused by false negatives being generated by an incorrect level of generalization. In general, the increasing number of participants brings more alert messages to the CIDS, consequently more false negatives are created. As a result, the detection rate decreases. Figure 6(b) shows the false positive rate of the proposed load balancing scheme compared to the previous source address correlation scheme. In general, our proposed load balancing scheme can achieve a false positive rate of less than 0.12% as shown in Figure 6(b). There is a slight increase in the detection rate from 0.03% to 0.12% when we increase the number of participants from 16 to 128, which may be caused by false positives being generated by an incorrect level of generalization. The increase in the number of participants results in more alert messages to the CIDS, consequently more false positives are created. As a result, the false positive rate increases.

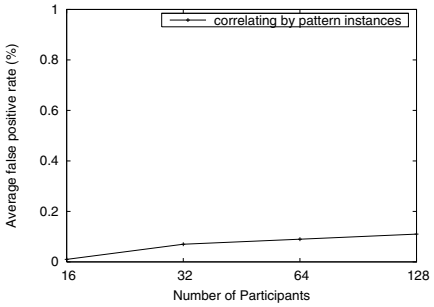
In this evaluation, we test for an exact match when we calculate the detection accuracy. We can observe fewer false positives and false negatives if we conduct a prefix match, such as considering two pattern instances as belonging to the same instance if they come from the same source. However, even using this strict measurement of detection accuracy, our load balancing scheme still achieves more than 90% detection accuracy and a false positive rate of less than 0.12%.

D. Simulation Results in the Real World Scan Scenario

In the second experiment, we first run our proposed approach over the 2:00-2:20am 2 January 2005 DShield dataset to measure the load distribution of our proposed scheme during the load surge observed in the normal scan scenario. Figure 7 plots the number of subscription messages on the top ten most heavily loaded participants among all the 810 participants. The pattern correlation scheme outperforms the source correlation scheme in terms of distributing the subscription messages among all participants. As shown in Figure 7, the number of subscription messages on the most heavily loaded participant using the pattern correlation scheme is half the number of messages on the counterpart participant using the source correlation scheme. The advantages of using our pattern correlation scheme is not as significant as the one shown in the simulation. This is because the top source accounts for fewer accesses (17% of the total accesses) than the top source in the simulation (50% of the total accesses).



(a) Detection rate of pattern correlation scheme



(b) False positive rate of pattern correlation scheme

Fig. 8. Evaluation of detection accuracy of the pattern correlation scheme in the normal scan scenario

In order to measure the detection accuracy of our proposed load balancing scheme in the normal scan scenario, we applied our proposed approach to a randomly selected date in the DShield dataset. We varied the number of participants from 16 to 128, and in each case, averaged the results over six different 30 minute time intervals. Figure 8 plots the evaluation results of detection accuracy. Figure 8(a) shows the detection rate of the proposed load balancing scheme compared to the previous source address correlation scheme, which exhibits the same trend as in Figure 6. In general, the detection rate of our pattern instance based correlation scheme achieves a detection rate of more than 92%. There is a slight decrease in the detection rate from 100% to 92% when we increase the number of participants from 16 to 128. This is caused by false negatives arising from alerts with the incorrect level of generalization. In general, the increasing number of participants brings more alert messages to the CIDS. Consequently more false negatives are created. As a result, the detection rate decreases. Figure 8(b) shows the false positive rate of the proposed load balancing scheme compared to the previous source address correlation scheme. In general, the proposed load balancing scheme can achieve a false positive rates of less than 0.11% as shown in Figure 8(b). There is a slight increase in the false positive rate from 0.01% to 0.11% when we increase the number of participants from 16 to 128, which is caused by the false positives generated at an incorrect level of generalization. The increasing number of participants brings more alert messages to the CIDS, consequently more false positives are created. As a result, the false positive rate increases.

In summary, the proposed load balancing scheme is able to evenly distribute the workload among participating nodes

while achieving a high detection rate (more than 90%) and negligible false positive rate (less than 0.12%) during worm outbreaks compared with the source address based correlation scheme. During the stealthy scan scenario, the proposed approach achieves more than 90% detection rate and less than 0.11% false positive rate compared with the source address based correlation scheme.

V. Related Work

In this section, we briefly review the related work in the field of decentralized collaborative intrusion detection. In order to address the detection challenge posed by large scale coordinated intrusions such as worms, DDoS attacks and coordinated scans, many collaborative systems have been proposed. Although load balancing is a crucial issue for any decentralized CIDS, this issue has been given a much lower priority than other design considerations, such as alert data sharing [7], [9] and alert data analysis [12].

The DOMINO project [8] is a CIDS that aims to monitor Internet-scale outbreaks. DOMINO uses three types of participants: *axis overlay, satellite communities* and *terrestrial contributors*, which monitor the Internet from different perspectives. The Lambda language [13] is used to aggregate the "raw alerts" from each firewall/NIDS locally. Then sampled alerts are correlated by creating an explicit mapping from the source address of the alert data to particular correlation nodes by using a hash function. Simple addition or averaging across each dimension of the data is used as a strategy for global aggregation. We propose a large scale intrusion detection system based on a P2P architecture in [3], [4]. Each participating IDS monitors the incoming traffic of its own subnetwork, and at each monitoring time interval Δ, it subscribes the suspicious information to the system using the source address as a key. The subscribers will then be notified if the suspicious evidence they subscribed to has been confirmed as an alert. The shortcoming of this type of source address based CIDS correlation scheme is that particular nodes become load hotspots during worm outbreaks, when increasing amounts of suspicious information are assigned to the same node.

Locasto *et al* [2] propose a P2P based CIDS, which uses Bloom filters [14] for preserving information privacy, and a dynamic overlay network for distributed correlation. This work addresses the load balancing issue by using a distributed correlation scheduling algorithm. This scheduling strategy improves the system performance in terms of bandwidth usage, but the trade-off between bandwidth usage and the level of false negatives introduced is not clear. I4 [6] is a P2P based inter-domain CIDS to reduce the amount of illegal traffic such as DDoS packets by sharing knowledge across different ISPs. However, the load balancing issue is not addressed in the above CIDSs.

VI. Conclusion

While CIDSs can help improve the accuracy and scalability of intrusion detection for many types of attacks, these distributed systems are potentially vulnerable to a severe load imbalance under certain common attack conditions, which can disrupt the performance of the CIDS. We propose a pattern instance based correlation scheme to automatically distribute the workload in a CIDS during worm outbreaks. Our simulation results show that our proposed approach is able to evenly distribute the workload among participant nodes while achieving a high detection rate and negligible false positive rate during worm outbreaks, compared with an earlier source address based correlation scheme.

Acknowledgment

We thank the NICTA Victoria Research Laboratory for funding this work. We would also like to thank the Internet Storm Center for providing us with the DShield Dataset, and Vinod Yegneswaran from SRI International for his help with the DShield Dataset.

References

[1] C. Kruegel, T. Toth, and C. Kerer, "Decentralized Event Correlation for Intrusion Detection," in *International Conference on Information Security and Cryptology (ICISC)*, 2001.

[2] M. Locasto, J. Parekh, A. Keromytis, and S. Stolfo, "Towards Collaborative Security and P2P Intrusion Detection," in *Proceedings of the 2005 IEEE Workshop on Information Assurance and Security*, 2005.

[3] C. V. Zhou, S. Karunasekera, and C. Leckie, "A Peer-to-Peer Collaborative Intrusion Detection System," in *IEEE International Conference on Networks (ICON), Malaysia*, 2005, pp. 118–123.

[4] ——, "Evaluation of a Decentralized Architecture for Large Scale Collaborative Intrusion Detection," in *the Tenth IFIP/IEEE International Symposium on Integrated Network Management (IM), Germany*, 2007, pp. 80–89.

[5] C. V. Zhou, C. Leckie, and S. Karunasekera, "Decentralized two-stage multi-dimensional alert correlation for collaborative intrusion detection," Department of Computer Science and Software Engineering, The University of Melbourne, Tech. Rep., 2007, http://www.cs.mu.oz.au/∼cvzhou/pub/data-agg.pdf.

[6] F. Zhao, V. R. Vemuri, S. F. Wu, F. Xue, and S. J. B. Yoo, "Interactive Informatics on Internet Infrastructure," in *the Tenth IFIP/IEEE International Symposium on Integrated Network Management (IM), Germany*, 2007, pp. 90–99.

[7] D. Dash, B. Kveton, J. Agosta, E. Schooler, J. Chandrashekar, A. Bachrach, and A. Newman, "When gossip is good: Distributed probabilistic inference for detection of slow network intrusions," in *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI)*, 2006.

[8] V. Yegneswaran, P. Barford, and S. Jha, "Global Intrusion Detection in the DOMINO Overlay System," in *Proceedings of Network and Distributed Security Symposium (NDSS)*, 2004.

[9] J. Garcia, F. Autrel, J. Borrell, S. Castillo, F. Cuppens, and G. Navarro, "Decentralized Publish-Subscribe System to Prevent Coordinated Attacks via Alert Correlation," in *Sixth International Conference on Information and Communications Security, October*, 2004, pp. 223–235.

[10] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web," in *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, 1997, pp. 654–663.

[11] Internet Storm Center, "Dshield org," http://www.dshield.org.

[12] A. Valdes and K. Skinner, "Probabilistic alert correlation," in *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID)*, vol. 68, 2001.

[13] F. Cuppens and R. Ortalo, "Lambda: A language to model a database for detection of attacks," in *Proceedings of Recent Advances in Intrusion Detection (RAID)*, 2000, pp. 197–216.

[14] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.